AERO：面向能量采集型物联网的自适应、高效、运行时感知 OTA 更新

Wei Wei*、Jingye Xu*、Sahidul Islam†、Dakai Zhu*、Chen Pan‡、Mimi Xie*
*美国德克萨斯州圣安东尼奥市，德克萨斯大学圣安东尼奥分校计算机科学系
†美国佐治亚州玛丽埃塔市，肯尼索州立大学计算机科学系
‡美国德克萨斯州圣安东尼奥市，德克萨斯大学圣安东尼奥分校电气与计算机工程系 电子邮箱：{wei.wei2,jingye.xu,dakai.zhu,chen.pan, mimi.xie}@utsa.edu,sislam19@kennesaw.edu

arXiv:2601.16935v1 [cs.AR] 23 Jan 2026

摘要——能量采集（EH）物联网（IoT）设备在间歇性能量供应条件下运行，这会干扰任务执行，使高能耗的空中（OTA）更新尤具挑战性。传统 OTA 更新机制依赖重启，会产生显著开销，因此不适用于间歇供电系统。近期的实时 OTA 更新技术降低了重启开销，但在更新与运行时执行交互时，仍缺乏确保一致性的机制。本文提出 AERO，这是一种自适应、高效的运行时感知 OTA 更新机制，可将更新任务集成到设备的有向无环图（DAG）中，并在能量和时序约束下与常规任务一同调度。AERO 通过识别受更新影响的执行区域并动态调整依赖关系，在适应间歇性能量供应的同时，确保更新的一致集成。在代表性工作负载上的实验表明，与现有实时更新方案相比，AERO 提高了更新的可靠性和效率。

索引词—**OTA 更新、能量采集、物联网、调度、有向无环图**

一、引言

对可持续、低维护计算的需求日益增长，加速了能量采集技术在下一代物联网设备中的应用。通过从太阳能、热能、动能和射频信号等环境能源中采集能量，能量采集设备 [1], 无需频繁更换电池，即可在资源受限的环境中自主运行。这些能力使能量采集系统特别适用于农业传感[2]、医疗植入设备 [3], 、野生动物追踪 [4], 和灾后恢复 [5]等应用。据预测，到 2028 [6], ，能量采集技术的全球市场规模将达到 65 亿美元，凸显了其在未来可持续物联网基础设施中的重要作用。

为确保能量采集设备的长期功能、安全性和适应能力，OTA 更新为能量采集物联网系统中的固件维护提供了一种前景广阔的机制。然而，频繁的电源中断会造成不可预测的执行窗口，使得安全、高效地完成 OTA 更新变得困难。传统的 OTA 更新机制[7], [8] 通常依赖重启至引导加载程序来应用固件更新。在此过程中，更新数据包会在正常执行期间接收并存储在非易失性存储器中，之后系统重启以安装更新。在内存有限的设备上，更新通常必须分成较小的数据块处理，从而导致多次重启。每次重启都会产生大量能耗开销、增加延迟并中断常规执行。对于间歇供电的能量采集设备，这些累积成本会显著延长停机时间，并增加更新不完整或失败的风险。

实时 OTA 更新技术能够在运行时应用更新，而无需重启并进入引导加载程序，从而提升执行的连续性 [9]。尽管这种方法减少了停机时间并维持了系统可用性，但也带来了新的正确性挑战。在实时更新期间，已更新和未更新的代码区域可能会暂时共存，易失性存储器中的残留状态也可能与新逻辑不一致。这种跨版本执行可能损害正确性和稳定性，因此需要采取机制，确保更新仅在安全的执行点应用。

尽管近期已有针对能量采集物联网设备提出更先进的实时 OTA 更新技术， [10], 但这些技术在执行期间仍无法充分确保正确性。现有方法要么忽视运行时出现的一致性问题，要么通过将更新推迟到所有常规任务完成后再进行来规避这些问题。这一局限表明，我们需要一种能够明确考虑执行状态的运行时感知 OTA 更新机制。

为克服这些局限，我们提出 AERO——一种面向能量采集物联网设备的运行时感知 OTA 更新机制。主要贡献总结如下：

- 我们提出一种轻量级 OTA 数据包格式，用于编码更新任务关系、常规任务依赖关系，以及针对更新代码和有向无环图的更新操作。
- 我们正式定义相互依赖的更新组，并用它们在有向无环图中确定相应的更新影响块，以刻画更新的一致性要求。
- 我们设计了一种运行时有向无环图调整算法，可在不同更新场景下将更新任务及其依赖关系与常规任务整合，从而在更新期间实现协调执行。
- 我们提出一种统一调度方法，在间歇性能量供应和截止期限约束下，对更新任务和常规任务进行联合调度。

II. 背景与动机

A. 背景

多项研究 [7], [8], [10]–[13] 探讨了面向资源受限物联网设备的 OTA 更新技术，其中许多

适用于能量采集物联网设备。这方面的大量研究通过缩小更新规模、提高代码相似性和优化编码，降低传输开销和能耗开销。其他研究则采用检查点策略 [14]–[16] 来容忍电源中断，或使用硬件感知方法 [17]–[21]，使更新适应可用的能量和内存约束。

为介绍我们的方法，本文回顾三类具有代表性的 OTA 更新技术：完整镜像更新、增量更新和实时更新。

1）完整镜像更新：完整镜像更新通过将整个固件镜像传输至设备并覆盖现有镜像来完成更新。MSP430FRBoot [22] 采用双闪存区实现这一方法：新镜像先写入非活动区，完成后再将其激活。Light Flash Write [19] 通过降低闪存写入的能耗，提高了间歇性供电下的可靠性。然而，对于能量采集物联网设备而言，完整镜像更新仍然代价高昂，因为传输并写入整个固件镜像需要大量能量和内存资源，在间歇执行期间更容易导致更新未完成。

2）增量更新：增量更新采用差分编码 [8]，只传输固件中经过修改的部分，从而降低开销。对于基于闪存的设备，基于分段的设计可在间歇式更新期间尽量减少内存操作和能耗 [17]。在网络层面，组播和波束成形可提高 LoRa 网络中多个设备的传输效率 [20], [21]。尽管具有这些优势，增量更新仍依赖于重启进入引导加载程序，这对间歇性供电的设备而言是个问题。

3）实时更新：实时更新可在运行时修改固件，无需重启。基于跳板的补丁技术 [9]引入了原子代码替换，近期的研究则将这一能力扩展到了间歇供电设备 [10]。尽管这些技术减少了重启开销，但它们通常假设更新可以安全应用且不会造成不一致，而这一假设在实时系统或间歇供电系统中并非总是成立。

B. 动机

尽管实时 OTA 更新减少了重启开销，但也带来了新的 一致性风险，这凸显了对运行时感知的 OTA 更新机制的需求；此类机制应明确管理更新任务与正在进行的执行之间的交互。在运行时修改期间，已更新和未更新的代码区域可能会暂时共存，从而导致混合版本执行。在某些情况下，这不会产生可观察到的影响；但在另一些情况下，则可能导致计算错误、执行失败或意外交互，如图 1 所示。在能量采集设备中，这些风险会进一步加剧，因为固件会被拆分为细粒度任务，以适应短电源周期，有时每个周期仅持续几个指令的时间 [23], [24]。在这种情况下，即使是细微的不一致，也可能引发多种故障类型。尽管先前的研究 [10]承认了这些问题，但如何处理混合版本执行这一挑战仍未解决。

进一步的动机在于观察到，更新任务无需严格等到常规任务完成，

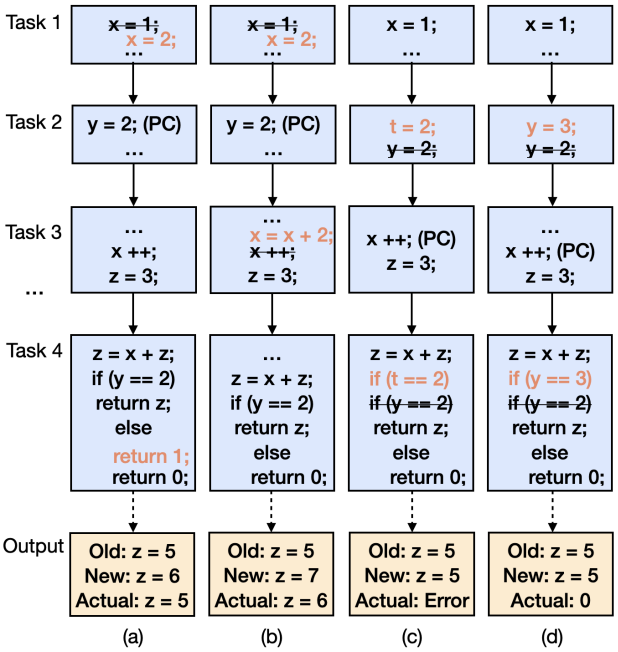

图 1. 混合版本执行的示例：（a）无影响，（b）计算错误，（c）执行失败，（d）非预期交互。

常规任务也不应一直停滞到更新完成。能量采集设备中的任务执行通常会因调度间隙而包含空闲时段，在此期间更新任务可以安全地执行。通过将更新视为有向无环图中的可调度实体，可以有效利用这些空闲时间推进更新，而不会干扰常规执行。

除了调度方面的考虑之外，更新还可能需要对有向无环图本身进行结构性更改。某些更新会修改任务或依赖关系、添加新任务，或删除过时任务，从而使有向无环图发生变化，而不仅仅是进行简单的代码替换。据我们所知，现有的 OTA 更新方法中，没有一种能够支持能量采集物联网设备上的此类有向无环图级更新。

III. AERO：运行时感知 OTA 更新机制

我们设计了 AERO，利用轻量级的依赖驱动数据包格式，将更新任务及其依赖关系集成到运行时有向无环图中。AERO 先进行运行时有向无环图调整，再进行统一调度，以便在能量和时序约束下协调更新任务与常规任务。AERO 强制执行任务级控制流和依赖关系一致性，并假设任务封装局部状态，且仅在任务边界访问共享内存、外设和中断。

A. 定义

1）相互依赖更新组定义：令$T = T_{\text{old}} \cup T_{\text{new}}$ 表示所有固件任务，其中$T_{\text{old}} = \{t_1, t_2, \ldots, t_m\}$ 为现有任务，$T_{\text{new}} = \{t_{m+1}, t_{m+2}, \ldots, t_n\}$ 为更新引入的新任务。执行过程建模为有向无环图 $G = (T, E)$，其中 $E \subseteq T \times T$和$(t_i, t_j) \in E$ 表示 t_j 依赖于 t_i。相应地，令 $U = U_{\text{old}} \cup U_{\text{new}}$ 表示更新任务集，其中每个 $u_i \in U$ 都与其对应的 $t_i \in T$相关联。更新任务会修改其相关任务的代码以及所有

关联依赖关系。如果两个更新任务必须在同一过程中更新才能保证正确性，则更新任务 u_i 依赖于另一个更新任务 u_j。相互依赖更新组是一个最小子集 $M \subseteq U$，其中每个 $u_i \in M$ 都至少依赖于一个$u_j \in M$，且只有当M 中的所有任务一同更新时，才能保证正确性。

2) 更新影响块定义：更新影响块是有向无环图中包含相互依赖更新组内所有任务，以及连接这些任务的依赖路径上的中间节点和边的最小子图。该块表示受更新影响的执行区域，集成时必须将其作为整体处理。

B. 依赖驱动更新数据包

AERO 引入了一种依赖驱动更新数据包格式，如图 2 所示，以便在能量采集物联网设备中实现运行时感知的更新集成。每个数据包载荷由一个组字段和一个更新操作组成。组字段仅包含在第一个数据包中，其中每个有向无环图常规任务对应一位，用于编码相互依赖更新组；1 表示该任务与更新相关联，0 表示不相关。未使用的位置保留供未来任务使用。该字段使系统能够及早确定更新影响块，并支持增量处理，因此即使内存无法一次容纳整个更新，更新也能继续进行。

更新操作包含一个头部和一个更新块。头部以一个 2 位操作码开头，用于指定更新类型，其依据是 [10] 中的间歇感知更新操作。1 位有向无环图标志用于指示更新块是否携带依赖信息。若该标志置位，更新块的前 n 位构成一个可选依赖字段，其中每一位指明更新任务是否依赖于对应的常规任务。头部剩余位用于标识任务标识符，而更新块以一段 m 位代码段结束，用于传递更新内容。

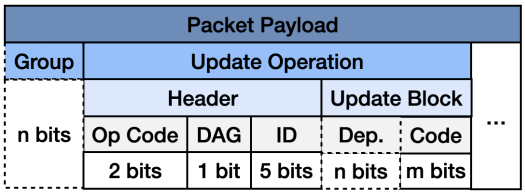

图2. AERO 更新数据包结构

C. 运行时有向无环图调整

为支持运行时感知更新而不中断执行，AERO 引入了一条专用更新任务链，该任务链在收到更新通知后触发。如图3所示，该任务链将更新任务及其依赖关系调整插入有向无环图中。

1) 修改现有任务：修改现有任务时，框架支持更新任务及其依赖关系。在图 4(a) 的示例中，更新任务修改t_4 和 t_8。集成取决于当前执行。如果运行时位于受更新影响的代码块内（例如，位于 t_4、 t_5、 t_6或t_8），则会延迟更新，直到相应任务完成

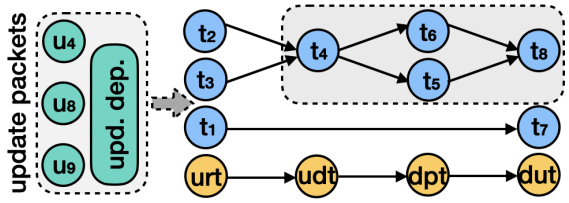

图3. AERO 更新流程概述。更新接收任务 (urt)：收集传入的数据包；更新解码任务 (udt)：解码数据包并重建更新任务；依赖处理任务 (dpt)：识别更新影响块；有向无环图更新任务 (dut)：插入更新任务并调整 G中受影响的依赖关系。

因此当前周期会在旧版本下完成。否则，更新会插入到关联任务之前，系统会在同一次执行中采用新版本。为清楚起见，此示例仅展示任务更新，不涉及依赖关系变更。

2) 插入新任务：该框架还支持向有向无环图中插入新任务。在图4(b)中，更新任务 u_9 创建了一个新任务 t_9。由于 t_9 事先并不存在，因此无论运行时如何，u_9 都必须先执行，以确保该任务可供调度。实际上，新任务可能会引入与现有任务的依赖关系，因此需要联合更新以保持一致性。为清楚起见，此示例仅展示了插入 t_9 ，不涉及依赖关系变更。

3) 移除过时任务： 该框架还支持从有向无环图中移除过时任务。在图4(c)中，更新任务u_8 移除任务 t_8 及其所有关联的依赖关系。如果运行时位于更新影响块内，则会将移除操作延迟至 t_8完成；否则会在 t_8 开始前执行。实际上，一个任务可能与其他任务存在依赖关系，移除该任务需要更新这些任务以保持一致性。为清晰起见，本示例仅展示移除 t_8 的情况，不涉及额外的依赖关系变更。

4) 运行时有向无环图调整算法： 基于上述案例，我们现在介绍 AERO 针对能量采集物联网设备的运行时有向无环图调整算法。该过程首先插入一个虚拟起始节点 s ，并将其连接到所有原始源节点，从而确保即使更新影响块 从图入口开始，在进入该块之前，执行仍可被暂时阻止。随后确定该块 B ，并移除所有进入 B 的边，以防止更新待处理期间产生新的执行路径。当内存无法容纳所有更新数据包时，这种阻塞至关重要，因为如果没有该机制， B 内部的常规任务可能会在过时代码下执行，从而导致不一致。通过将进入 B 的操作延迟到更新任务完成之后，AERO 在允许该块之外不受影响的任务继续执行的同时，保证了正确性。遵循先前的间歇式更新设计， [10],AERO 会将更新任务和常规任务都检查点保存到非易失性存储器中，使部分应用的更新能够在断电后安全恢复并重新应用。

D. 统一调度算法

集成更新任务后，AERO 采用统一调度器来协调常规任务和更新任务，同时适应

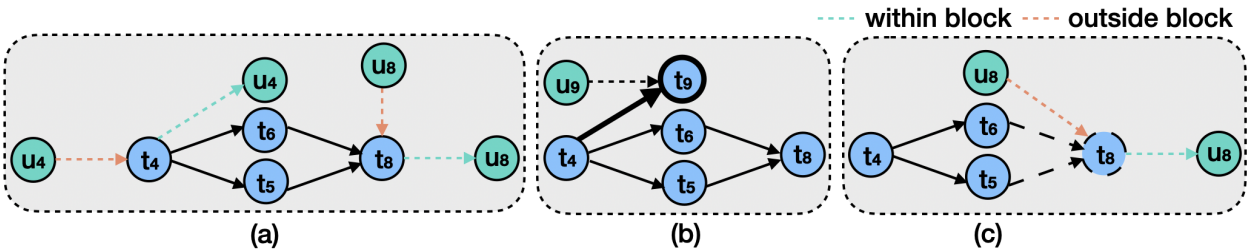

图 4. AERO 更新案例：(a) 修改现有任务 t_4 和 t_8；(b) 插入新任务 t_9；(c) 移除过时任务 t_8

算法 1运行时有向无环图调整算法

输入： $G = (T_{\text{old}},\ E)$；更新组 M；当前任务 t_{curr}
输出： 修改后的有向无环图 G'，阻塞边 E_{blk}
$T' \leftarrow T_{\text{old}} \cup \{s\}$, $E' \leftarrow E \cup \{(s, v) \mid v \in \text{来源}(T_{\text{old}})\}$ // 添加虚拟起始节点 s
$B \leftarrow f(M,\ (T',\ E'))$ // 识别更新影响块
$E_{\text{blk}} \leftarrow \{(x,\ y) \in E' \mid y \in B\}$ // 备份阻塞边
$E' \leftarrow E' \setminus E_{\text{blk}}$ // 阻止进入 B
对于每个 $u_i \in M$ **执行**
令 t_i 为与 u_i 相关联的任务
如果 $t_i \notin T'$，**则**
$T' \leftarrow T' \cup \{u_i, t_i\}$ // 添加 u_i 和占位符 t_i
$E' \leftarrow E' \cup \{(u_i, t_i)\}$ // 添加边 (u_i, t_i)
else
if $t_{\text{curr}} \in B$ **然后**
$T' \leftarrow T' \cup \{u_i\}$
$E' \leftarrow E' \cup \{(t_i, u_i)\}$ // 延迟至 t_i 完成
else
$T' \leftarrow T' \cup \{u_i\}$
$E' \leftarrow E' \cup \{(u_i, t_i)\}$ // 在 t_i之前更新
结束如果
结束如果
结束循环
$G' \leftarrow (T',\ E')$
返回 G', E_{blk}

能量和截止期限约束。图5以图形方式概述了任务选择过程，而算法2展示了详细流程。二者共同说明了如何安全高效地应用更新，同时不影响实时性能，也不超出能量预算。

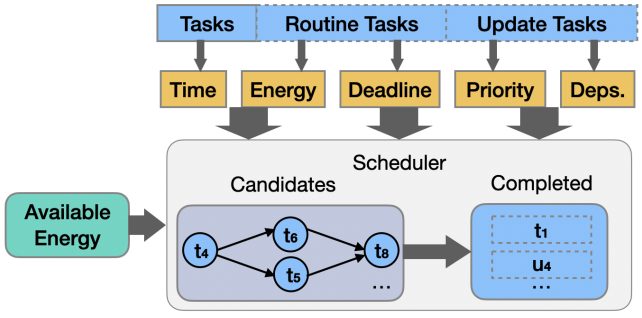

图5. 任务选择过程示意图

任务选择通过优先级队列处理，队列按截止期限排序，其次按优先级排序。常规任务直接执行，而更新任务则根据其操作进行应用。如果更新操作中的 DAG 位已设置，则相应的依赖边会在应用更新之前添加到有向无环图中。当当前组 M 中的所有更新任务完成后，调度器会恢复算法1中移除的受更新影响的区块 B 的入口边。随后，通过移除更新任务、辅助任务、虚拟起始节点及其连接边来清理图。完整流程概述见算法2。

算法2统一调度算法

输入： $G' = (T',\ E')$；阻塞边 E_{blk}
输出： 每个调度步骤中执行的任务 t^*
$Q \leftarrow \emptyset$ // 按截止期限排序的优先级队列→优先级
当真 执行
对于每个 $t \in T'$ with $(\forall p \in \text{pred}(t)$: p 已完成$)$ **执行**
$Q \leftarrow Q \cup \{t\}$
结束循环
如果 $Q = \emptyset$ **则**
继续 // 无可调度任务，重新检查
else
$t^* \leftarrow$弹出(Q) // 最早截止期限优先，其次是优先级
如果 $t^* \in T$ **则**
执行(t^*) // 执行常规任务
else
如果 头部(t^*)。有向无环图 $= 1$ **则**
$E' \leftarrow E' \cup \Delta E(t^*)$ // 添加新边
结束 if
apply(t^*) // 应用更新操作
结束 if
状态$(t^*) \leftarrow$ 已完成
结束如果
如果 $M \subseteq \{$已完成任务$\}$ **则**
$E' \leftarrow E' \cup E_{\text{blk}}$ // 将阻塞边恢复到 B
$S \leftarrow M \cup \{urt, udt, dpt, dut, s\}$
$T' \leftarrow T' \setminus S$ // 移除更新、辅助节点、虚拟节点
$E' \leftarrow E' \setminus \{(u,\ v) \in E' \mid (u \in S) \vee (v \in S)\}$
$G \leftarrow (T',\ E')$ // 存储更新后的有向无环图
跳出
结束 if
结束 while

IV. 实验

A. 实验设置

所有实验均在 TI MSP430FR5994 微控制器上进行 [25]。这是一款资源受限的设备，具有超低功耗运行能力并内置铁电随机存取存储器，因此适用于能量采集物联网系统。使用 CCS（Code Composer Studio）中的 TIEnergyTrace [26] [27]对基准任务进行独立性能分析，从而获得执行时间和能耗。更新任务的执行时间和能耗

数值根据各自的更新规模确定。随后使用这些分析结果，评估 AERO 统一调度下的任务执行和更新性能。

B. 基准测试设计

为评估 AERO 的有效性，我们采用四个基准测试，代表不同的应用类型和 DAG 结构。对于每个基准测试，电容器容量均设置为恰好足以执行能量需求最高的任务。该配置使系统在接近能量极限的状态下运行，为评估运行时感知更新提供了真实条件。表 I 总结了这些基准测试及其配置。

这四个基准测试涵盖了与 AERO 相关的运行时感知更新的不同方面。快速排序（B1） [28] 展示了包含极小任务的细粒度调度。AES 加密（B2） [15] 考察了跨并行但功能各异的硬件和软件执行路径的更新行为。LeNet-5（B3） [29] 提供了严格的线性任务流水线，适用于分析有序执行下的更新传播。心率监测器（B4） [30] 在分叉-汇合 DAG 中结合了并行处理与重新合并，可用于评估部分并行执行过程中的一致性。

C. 空中更新场景

这些基准共同为评估不同应用类型和 DAG 结构中的运行时感知更新提供了依据。在此基础上，我们根据表 I 中的基准设计了六种更新场景。这些场景涵盖两类更新。功能修改会改变任务行为，例如调整排序例程、替换神经网络的最后阶段，或引入新的信号处理步骤。参数修改则会改变配置参数而不影响任务行为，例如增大 AES 密钥或块大小、更新模型权重，或更改 UART 设置。

每种场景都会引入一个或多个更新任务，形成相互依赖的更新组，必须在运行时集成这些任务，同时不违反 DAG 依赖关系或损害一致性。更新规模涵盖从小型任务级变更到能量采集物联网设备上的大型固件更新。表 II 总结了这六种场景及其更新规模。

我们没有为完整镜像更新或 DAG 更新单独定义场景。任何方法都可以执行完整镜像更新，因此它无法区分不同方法；而 DAG 更新是 AERO 独有的支持功能，并在第 V 节通过功能比较进行评估。

V. 评估

评估通过模拟进行，使用从独立任务分析中获取的执行时间和能耗值。可用能量根据来自 EH IoT 设备 [31]，的真实太阳能轨迹持续更新，更新到达则遵循伪随机过程，以模拟随机行为。任务优先级遵循有向无环图：较早的任务优先级较高，较晚的任务优先级较低，而更新任务始终最后运行。我们假设非易失性存储空间充足，并且所有更新数据包都在集成之前到达。虽然 AERO 可以处理多数据包的增量更新，但这一假设消除了通信成本这一变量，从而能够公平地比较运行时执行情况。作为比较，我们将 AERO 与两个具有代表性的基线进行评估：实时更新基线对应于 [9], 中的基于跳板的实时更新方法，而间歇式更新基线对应于 [10]中的检查点辅助 OTA 框架。

A. 功能比较

我们首先将 AERO 的功能与基线方案进行比较。基线方案具备成熟的能力，例如无需重启即可实时更新，以及通过一致性机制实现间歇执行；两者都支持渐进式完整映像更新。然而，两者均不支持运行时集成或 DAG 更新。AERO 将更新任务集成到 DAG 中，并与常规任务一同调度，从而实现现有方法无法支持的任务依赖关系更新。图 6 总结了这些差异。

Feature	Live	Intermittent Update	AERO
Live Update Support	✓	✓	✓
Progressive Full Image Update	✓	✓	✓
Intermittent Update Support	✗	✓	✓
Consistency Guarantee	✗	✓	✓
Runtime Integration	✗	✗	✓
DAG Update Support	✗	✗	✓

图 6. 更新方法的功能比较

B. 更新错误率

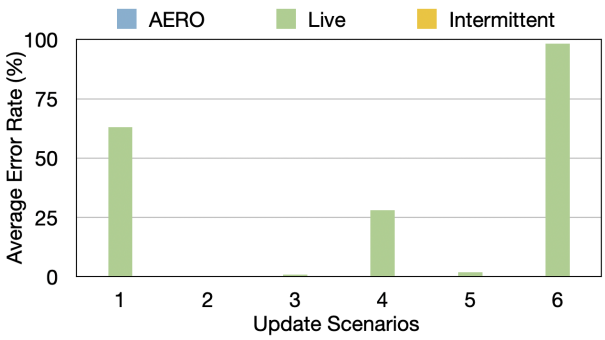

图 7. 平均错误率比较

如图 7 所示，AERO 在所有场景中均保持零错误率。相比之下，当更新发生在有向无环图（DAG）的计算密集区域时，实时更新方法经常会引入不一致；例如，在场景 4 和场景 6 中，更新与高成本任务重叠，导致场景 6 的错误率高达 98.3%。即使是由轻量级任务组成的场景 1，在实时更新下也会出现较高的错误率，因为执行时间普遍较短，增加了中断活动任务的可能性。相比之下，场景 2、3 和 5 表明，实时更新可以适用于小规模、独立的更新，但仍需采取额外的保护措施以确保正确性。间歇式更新也能保持零错误率

表 I 评估基准

基准	规模（B）	DAG 类型	容量（mF）	描述
B1：快速排序 [28]	1,696	线性	0.02	四次顺序快速排序操作
B2：AES 加密 [15]	3,975	并行	0.2	使用硬件/软件路径进行 AES 加密/解密
B3：LeNet-5 [29]	50,632	线性	10	使用优化后的 LeNet-5 模型进行推理
B4：心率监测器 [30]	46,050	分叉-汇合	1	基于光电容积描记法的心率估计

表 II 空中更新场景

ID	描述	规模（B）
1	B1：修改第二排序任务	280
2	B2：将 AES 密钥改为 192 位（acc/DMA）	130
3	B2：将 AES 分组改为 256 位	162
4	B3：将最终层改为全连接层	292
5	B4：添加传感器支持	702
6	B4：更新 HR 模型和 UART	6246

这是其设计所保证的，但代价是更新延迟较长，对于安全补丁等对时间敏感的更新而言，这种延迟可能无法接受。

C. 更新完成时间

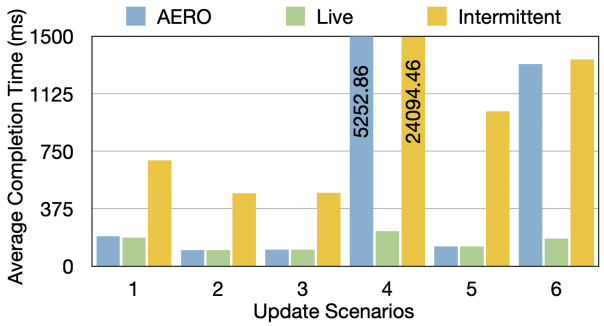

图 8. 平均更新完成时间对比

图 8 展示了所有方法的平均更新完成时间。AERO 在场景 1–5 中始终优于间歇式更新，因为它通过将更新任务直接集成到有向无环图中，更快地应用更新；即使在更新规模最大的场景 6 中，其完成时间仍略短于其他方法。与实时更新相比，我们的方法在场景 1、2、3 和 5 中的完成时间相近。只有在更新影响块 执行时才会出现小幅延迟，因为为保持正确性，更新会被推迟；这与图 7 中零错误率的结果一致。这种权衡在场景 4 和场景 6 中最为明显：实时更新完成得更快，但错误率非常高，如图 7 所示。

D. 截止期错失率 (DMR)

图 9 比较了三种方法的 DMR。仅为常规任务设定截止期限，设定值为分析得到的执行时间加上 0.5× 的裕量，以考虑调度开销；更新任务不设截止期限，且总是在常规任务之后执行。在所有

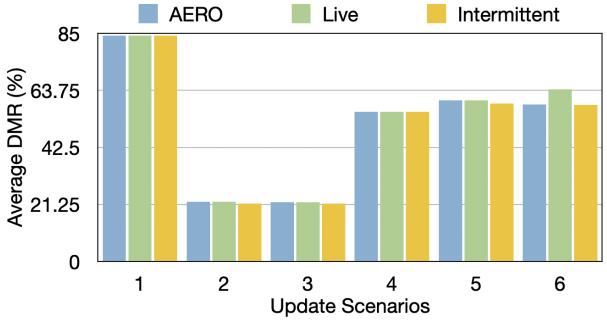

图 9. 平均 DMR 对比

场景中，AERO 的结果与间歇式更新相近或略高，表明更新可以在运行时集成，且不会产生过多开销。与实时更新相比，AERO 在场景 1–5 中的 DMR 相近。在场景 6 中，由于更新规模相对较大，AERO 的 DMR 较低；这是因为实时更新会立即应用更新，而不考虑当前的执行状态，从而延迟常规任务并增加 DMR。

尽管我们的实验使用了 MSP430FR5994 硬件和太阳能量轨迹，但 AERO 并不局限于这一平台或能源。其 DAG 级设计使其能够广泛适用于各种能量采集物联网设备和环境，同时仅在更新到达时触发运行时有向无环图调整，才会产生开销。该开销主要来自修改有向无环图时产生的非易失性存储器写入；在实际应用中，对于我们评估的场景而言，这一开销可以忽略不计，且不会影响观察到的评估趋势。

VI. 结论

本文介绍了 AERO，这是一种面向间歇供电物联网设备的运行时感知 OTA 更新机制。通过将更新任务集成到正在执行的有向无环图中，AERO 在确保正确性的同时，支持设备在正常运行期间进行更新。使用真实世界能量轨迹和硬件配置文件进行的仿真表明，与实时与间歇基线相比，AERO 实现了零更新错误、更短的更新完成时间以及具有竞争力的 DMR。除了单设备评估之外，AERO 还为能量采集物联网网络中的协同更新研究提供了可能。总而言之，AERO 提供了一种实用且可靠的 OTA 更新解决方案，其 DAG 级设计可适用于不同的硬件平台和能量采集条件，从而支持可持续、长生命周期的物联网部署。

VII. 致谢

本研究部分得到了美国国家科学基金会资助，项目编号为 CNS-2443885 和 CNS-2318641。

参考文献

[1] Yuehang Sun、Yun-Ze Li 和 Man Yuan。关于可穿戴设备电源供应与能量采集的需求、挑战和新思路。*Nano Energy*，115:108715，2023年。[2] Kapil Aggarwal、G Sreenivasula Reddy、Ramesh Makala、T Srihari、Neetu Sharma 和 Charanjeet Singh。无线传感器网络农业监测节能技术研究。*Computers and Electrical Engineering*，113:109052，2024年。[3] Md Maruf Hossain Shuvo、Twisha Titirsha、Nazmul Amin 和 Syed Kamrul Islam。用于持久精准医疗的植入式和可穿戴医疗设备能量采集。*Energies*，15 7495，2022 20( ): 。[4] Sivan Toledo、Shai Mendel、Anat Levi、Yoni Vortman、Wiebke Ullmann、Lena-Rosa Scherer、Jan Pufelski、Frank Van Maarseveen、Bas Denissen、Allert Bijleveld 等。Vildehaye：一系列多功能、适用范围广且经实地验证的轻量级野生动物追踪与传感标签。载于*2022*年第*21*届*ACM/IEEE*传感器网络信息处理国际会议*(IPSN)*，第1–14页。IEEE，2022年。[5] Xiuchuan Sun、Jian Chen、Haoran Zhao、Wen Zhang 和 Yicheng Zhang。基于信息物理协同优化的韧性配电网序贯灾后恢复策略。*IEEE Transactions on SmartGrid*，14(2):1173–1187，2022年。[6] 能量采集的全球市场、技术和设备。https://www.bccresearch.com/market-research/energy-and-resources/global-markets-technologies-and-devices-for-energy-harvesting.html#:~:text=What%20is%20the%20projected%20market,11.4%25%20during%20the%20forecast%20period, 2023年7月。[7] Wei Dong、Yunhao Liu、Chun Chen、Jiajun Bu、Chao Huang 和 Zhiwei Zhao。R2：在联网嵌入式系统中使用可重定位代码进行增量式重新编程。*IEEE Transactions on Computers*，62(9):1837–1849，2012年。[8] Wei Wei、Sahidul Islam、Jishnu Banerjee、Shanglin Zhou、Chen Pan、Caiwen Ding 和 Mimi Xie。一种用于更新能量采集设备上深度学习权重的间歇式OTA方法。载于*2022*年第*23*届电子设计质量国际研讨会*(ISQED)*，第1–6页。IEEE，2022年。[9] Chi Zhang、Wonsun Ahn、Youtao Zhang 和 Bruce R Childers。能量采集环境中物联网设备的在线代码更新。载于*2016*年第*5*届非易失性存储器系统与应用研讨会*(NVMSA)*，第1–6页。IEEE，2016年。[10] Wei Wei、Chen Pan、Sahidul Islam、Jishnu Banerjee、Shyamala Palanisamy 和 Mimi Xie。面向微型能量采集设备的间歇式OTA代码更新框架。*IEEE Transactions on Computer-Aided Design of IntegratedCircuits and Systems*，2024年。[11] Joel Koshy 和 Raju Pandey。用于传感器网络节能重新编程的远程增量链接。载于第二届欧洲无线传感器网络研讨会论文集，*2005*年。，第354–365页。IEEE，2005年。[12] Chih-Chieh Han、Ram Kumar Rengaswamy、Roy Shea、Eddie Kohler 和 Mani Srivastava。SOS：面向传感器网络的动态操作系统。载于第三届移动系统、应用与服务国际会议论文集*(Mobisys)*，第10卷，第1067170–1067188页。Citeseer，2005年。[13] Rajesh Krishna Panta 和 Saurabh Bagchi。Hermes：面向无线传感器网络的快速节能增量代码更新。载于*IEEE INFOCOM 2009*，第639–647页。IEEE，2009年。[14] Priyanka Singla 和 Smruti R Sarangi。能量采集设备检查点技术的综述与实验分析。*Journal of Systems Architecture*，126:102464，2022年。[15] Shyamala Palanisamy、Wei Wei 和 Mimi Xie。面向能量采集设备的节能、持久安全的基于块的差分检查点技术。载于*2025*年第*26*届电子设计质量国际研讨会*(ISQED)*，第1–6页。IEEE，2025年。[16] Sahidul Islam、Wei Wei、Jishnu Banerjee 和 Chen Pan。面向低功耗能量采集系统的能量自适应无检查点间歇推理。载于*2025*年第*26*届电子设计质量国际研讨会*(ISQED)*，第1–7页。IEEE，2025年。[17] Wei Wei、Jishnu Banerjee、Sahidul Islam、Chen Pan 和 Mimi Xie。面向基于闪存的无电池物联网设备的能量感知增量OTA更新。载于*2024*年*IEEE*计算机学会*VLSI*年度研讨会*(ISVLSI)*，第51–56页。IEEE，2024年。[18] Jishnu Banerjee、Sahidul Islam、Wei Wei、Chen Pan、Dakai Zhu 和 Mimi Xie。面向物联网设备的内存感知高效深度学习机制。载于*2021*年第*32*届*IEEE*应用国际会议-系统、架构与处理器（*ASAP*），第 187–194 页。IEEE，2021 年。[19] Songran Liu、Mingsong Lv、Wei Zhang、Xu Jiang、Chuancai Gu、Tao Yang、Wang Yi 和 Nan Guan。面向能量采集物联网设备高效固件更新的 Light Flash Write。载于 *2023* 年欧洲设计、自动化与测试大会暨展览会（*DATE*），第 1–6 页。IEEE，2023 年。[20] Zehua Sun、Tao Ni、Huanqi Yang、Kai Liu、Yu Zhang、Tao Gu 和 Weitao Xu。Flora+：能效高、可靠、波束成形辅助且安全的 LoRa 网络空中固件更新。*ACM* 传感器网络汇刊，20(3)：1–28，2024 年。[21] Sukanya Jewsakul 和 Edith CH Ngai。Fiora：能量中和感知的能量采集 LoRa 网络组播固件分发。载于第 *11* 届 *ACM* 能源高效建筑、城市与交通系统国际会议论文集，第 88–98 页，2024 年。[22] Ryan Brown、Katie Pier 和 Gary Gao。MSP430FRBoot：面向 MSP430 FRAM 大内存模型设备的主存引导加载程序和空中更新。https://www.ti.com/lit/an/slaa721e/slaa721e.pdf，2020 年 3 月。[23] Shuo Xu、Wei Zhang、Mengying Zhao、Zimeng Zhou 和 Lei Ju。面向高效间歇式计算系统的缓存感知任务分解。载于第 *61* 届 *ACM/IEEE* 设计自动化大会论文集，第 1–6 页，2024 年。[24] Hehe Li、Yongpan Liu、Chenchen Fu、Chun Jason Xue、Donglai Xiang、Jinshan Yue、Jinyang Li、Daming Zhang、Jingtong Hu 和 Huazhong Yang。考虑电源切换开销的能量采集非易失处理器性能感知任务调度。载于 第 *53* 届年度设计自动化大会论文集，第 1–6 页，2016 年。[25] 德州仪器。MSP430FR5994 微控制器。https://www.ti.com/product/MSP430FR5994，2024 年。[26] 德州仪器。EnergyTrace。https://software-dl.ti.com/ccs/esd/documents/xdsdebugprobes/emu_energytrace.html。[27] 德州仪器。Code Composer Studio。https://www.ti.com/tool/CCSTUDIO。[28] Matthew R Guthaus、Jeffrey S Ringenberg、Dan Ernst、Todd M Austin、Trevor Mudge 和 Richard B Brown。MiBench：免费且具有商业代表性的嵌入式基准测试套件。载于 第四届年度 *IEEE* 负载特征分析国际研讨会论文集。*WWC-4*（目录号：*01EX538*），第 3–14 页。IEEE，2001 年。[29] Sahidul Islam、Shanglin Zhou、Ran Ran、Yu-Fang Jin、Wujie Wen、Caiwen Ding 和 Mimi Xie。Eve：面向低功耗能量采集系统的环境自适应神经网络模型。载于 第 *41* 届 *IEEE/ACM* 计算机辅助设计国际会议论文集，第 1–9 页，2022 年。[30] Jingye Xu、Yuntong Zhang、Mimi Xie、Wei Wang 和 Dakai Zhu。利用光电容积脉搏波传感器对心率变异性进行实时智能端侧监测。*Journal of Systems Architecture*，154：103240，2024 年。[31] Jishnu Banerjee、Sahidul Islam、Wei Wei、Chen Pan 和 Mimi Xie。AutoTile：面向无电池嵌入式系统深度推理的自主任务分块。载于 *2024* 年五大湖超大规模集成电路研讨会论文集，第 323–327 页，2024 年。